2.1 Minimum Spanning Trees

Let $G = (V, E)$ be a connected undirected graph.

Definition 2.4 A *forest* in G is a subgraph $F = (V, E')$ with no cycles. Note that F has the same vertex set as G. A *spanning tree* in G is a forest with exactly one connected component. Given weights $w : E \to \mathcal{N}$ (edges are assigned weights over the natural numbers), a *minimum (weight) spanning tree (MST)* in G is a spanning tree T whose total weight (sum of the weights of the edges in T) is minimum over all spanning trees. □

Lemma 2.5 *Let $F = (V, E)$ be an undirected graph, c the number of connected components of F, $m = |E|$, and $n = |V|$. Then F has no cycles iff $c + m = n$.*

Proof.
($\rightarrow$) By induction on m. If $m = 0$, then there are n vertices and each forms a connected component, so $c = n$. If an edge is added without forming a cycle, then it must join two components. Thus m is increased by 1 and c is decreased by 1, so the equation $c + m = n$ is maintained.

($\leftarrow$) Suppose that F has at least one cycle. Pick an arbitrary cycle and remove an edge from that cycle. Then m decreases by 1, but c and n remain the same. Repeat until there are no more cycles. When done, the equation $c + m = n$ holds, by the preceding paragraph; but then it could not have held originally. □

We use a *greedy algorithm* to produce a minimum weight spanning tree. This algorithm is originally due to Kruskal [66].

Algorithm 2.6 (Greedy Algorithm for MST)

1. Sort the edges by weight.
2. For each edge on the list in order of increasing weight, include that edge in the spanning tree if it does not form a cycle with the edges already taken; otherwise discard it.

The algorithm can be halted as soon as $n - 1$ edges have been kept, since we know we have a spanning tree by Lemma 2.5.

Step 1 takes time $O(m \log m) = O(m \log n)$ using any one of a number of general sorting methods, but can be done faster in certain cases, for example if the weights are small integers so that bucket sort can be used.

Later on, we will give an almost linear time implementation of step 2, but for now we will settle for $O(n \log n)$. We will think of including an edge e in the spanning tree as taking the union of two disjoint sets of vertices, namely the vertices in the connected components of the two endpoints of e in the forest

being built. We represent each connected component as a linked list. Each list element points to the next element and has a back pointer to the head of the list. Initially there are no edges, so we have n lists, each containing one vertex. When a new edge (u, v) is encountered, we check whether it would form a cycle, *i.e.* whether u and v are in the same connected component, by comparing back pointers to see if u and v are on the same list. If not, we add (u, v) to the spanning tree and take the union of the two connected components by merging the two lists. Note that the lists are always disjoint, so we don't have to check for duplicates.

Checking whether u and v are in the same connected component takes constant time. Each merge of two lists could take as much as linear time, since we have to traverse one list and change the back pointers, and there are $n - 1$ merges; this will give $O(n^2)$ if we are not careful. However, if we maintain counters containing the size of each component and always merge the smaller into the larger, then each vertex can have its back pointer changed at most $\log n$ times, since each time the size of its component at least doubles. If we charge the change of a back pointer to the vertex itself, then there are at most $\log n$ changes per vertex, or at most $n \log n$ in all. Thus the total time for all list merges is $O(n \log n)$.

2.2 The Blue and Red Rules

Here is a more general approach encompassing most of the known algorithms for the MST problem. For details and references, see [100, Chapter 6], which proves the correctness of the greedy algorithm as a special case of this more general approach. In the next lecture, we will give an even more general treatment.

Let $G = (V, E)$ be an undirected connected graph with edge weights $w : E \to \mathcal{N}$. Consider the following two rules for coloring the edges of G, which Tarjan [100] calls the *blue rule* and the *red rule*:

> **Blue Rule:** Find a *cut* (a partition of V into two disjoint sets X and $V - X$) such that no blue edge crosses the cut. Pick an uncolored edge of minimum weight between X and $V - X$ and color it blue.
> **Red Rule:** Find a *cycle* (a path in G starting and ending at the same vertex) containing no red edge. Pick an uncolored edge of maximum weight on that cycle and color it red.

The greedy algorithm is just a repeated application of a special case of the blue rule. We will show next time:

Theorem 2.7 *Starting with all edges uncolored, if the blue and red rules are applied in arbitrary order until neither applies, then the final set of blue edges forms a minimum spanning tree.*